APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/769,535	01/30/2004	Milton E. Moskowitz	H0005134- -1623	8642

128 7590 04/21/2009
HONEYWELL INTERNATIONAL INC.
101 COLUMBIA ROAD
P O BOX 2245
MORRISTOWN, NJ 07962-2245

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
04/21/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/769,535	**Applicant(s)** MOSKOWITZ ET AL.
	Examiner TUAN A. VU	**Art Unit** 2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _22 January 2009_.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-18,21 and 22_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-18, 21-22_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 1/22/09.

As indicated in Applicant's response, claims 1-4, 7-9, 12, 15-17 have been amended. Claims 1-18, 21-22 are pending in the office action.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the first paragraph of 35 U.S.C. 112:

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

3. Claims 1-18, 21-22 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

As per claims 1, 12, the limitation recited as '*generating expected code* for the generated code' (cl. 1) or '*generate expected computer code*' (cl. 12) is not deemed justified by a corresponding description in the Disclosure. As disclosed, the verification module 102 applies generated code 103 and the information collected from parsing model_file 101 (Fig. 1). The information derived from reading the model are construed as input/output parameters or format configuration type of meta-information (see Fig. 3a; Specifications: *databases information ... format of the expected code for each block* -para 0027 pg. 7; see Table pg. 7; *specific information ... is determined* – para 0030 pg. 8; *configuration ... is stored for future use* - top, pg. 9) some of which are persisted as database (see para 0036, pg. 10) in terms of proper form of a command.

Database stored information as reference to support how a form of the intended target code should be implemented does not clearly depict a clearly actual step taken (or possessed by the inventor) whereby based on model, a "expected" code has been generated, so that this very "expected" computer code is then inputted into the verification module of Fig. 3. for the 'comparing' step as precisely recited in the claims (i.e. comparing each line … the generated code and the expected code…). The information gathered from the model file as mere meta-information or form-related references from database, all of which deemed not having line-by-line correspondence (emphasis added) to the generated code file 103 (Fig. 1) or Ccrdemo.c (Fig. 3b). Further step 404 (Fig. 4) only determines which part of model_file should have associated 'generated code', then step 410-414 checks whether the read Code_file has proper form or expected model requirements (see para 0035-0036, pg. 10), such verification of *code_file* against a model (see step 102, Fig. 1) not indicative that a clear action of <u>generating</u> 'expected code' (emphasis added) has been achieved or possessed by the inventor, on the basis of reading the model_file 101. The above language is deemed not enabled by or provided with specific description therefor in the Disclosure. The 'generate expected code' step will be treated as determining format or requirements from parsing or reading a model, and *each line* of such (generated) *expected code* (e.g. in relation to being compared to line of generated code) is not given patentable weight in the *comparing* step.

Claims 2-11, 21, 13-18, 22 for not remedying to the lack of Description as identified above, will be rejected for the same deficiency.

4. The *comparing* of line-by-line or header-against-header as recited in the above recited claim language is also rejected as non-enabled, as follows.

Claim 7 recites *compares each line* in the generated code and the expected code to determine … transmits an error message *if a first line* in the generated computer code … based on the comparison'. The generated expected code is not deemed being supported with sufficient if any teachings (see above Rejection) from scanning the Disclosure for one to make use of the claimed 'comparing each line' step in order to implement transmitting *a error message if a first line* of the generated code is not proper based on the comparing recited from above. The 'if a first line' limitation is nowhere disclosed in the comparison of *code_file* with reference to some required form obtained from model_file in Figure 4. That is, the generated *code file* (Fig. 3b) as depicted in the verification stage (Fig. 4) is validated against form requirements persisted in (i) database (para 0027, pg. 7) or (ii) via tagged constructs (para 0032) used as template guidelines for source code (*code_file 103*, Fig. 1) to be generated. Accordingly, no line of *expected computer code* (not *first line*, not *second line*) is being actually generated and subsequently matched against a corresponding line (first line, second line) of generated code like that of Ccrdemo.c (Fig. 3b) in the course of validation described in para 0036. The error message based on first line (of *generated code*) not conforming to any line of a *generated expected code* (having first line, second line of a tangible computer source code) will not be given patentable weight; and will be treated as message based on validation of generated code against a requirement for a form, syntax expected for a given line. Claims 8-11, 22 for reciting *first line*, or *second line* of *expected computer code*, fail to remedy to lack of support identified when addressing the above 'error message … if first line' in claim 7; and are rejected likewise.

As per claims 5, 11, 18, the 'comparing' limitation as to matching 'a first header information section in the generated code' with 'a second header information section in the expected code' is not found as having a clear and proper description in the Specifications. Scanning the Disclosure for 'header section' it is observed that Figure 3B pg. 4 depicts header section (or header file) of the generated code *Ccrdemo.c*, and paragraph 0032, pg. 9 describes how shorthand notation labels (<S4>) of a model are used as indicators based on which to be place generated code construct of the header file. Expected code is not disclosed as actually generated, as set forth in the above portions of this 112, 1st paragraph Rejection. Thus, shorthand labels being mere syntax indicators, place holder, or form helper cannot be construed as 'header information section' belonging to a complete source code file generated that would fit the claim language of *expected code being generated*. Nowhere in the Specifications is the 'expected code' disclosed as explicitly being generated and including a main body and an header section portion with line-by-line correspondence to *code_file 101* (Fig. 1) or *Ccrdemo.c* (Fig. 3b) ; and as described this "expected" code or header thereof amounts to mere meta-information (see line 342, Fig. 3B(Continued); para 0038, 0044, pg. 11) implemented as syntactic form using bracket and labels < > like tagged language or meta reference information persisted in database to support the comparing (see Fig. 4; para 0036, pg. 10). In short, tagged format or bracketted place holders (para 0032, pg. 9), which at best, serve as guidelines, cannot be equated to 'a second header information section' comprised in generated code form or "expected code" having line-by-line correspondence to code_file 101. The 'second header section' is not given any weight because it is deemed that the inventor does not possess this feature at the time the invention was made. The above 'comparing' step will be treated as though 'first header section' in the generated code is matched against expected code that is derived from meta-information from the model.

Claims 15-17 recite 'comparing each line in the expected code **to determine** if the plurality of lines ...', in the sense that the plurality of lines of generated code are in expected form(cl. 15) or include correct number of lines (cl. 16), or in proper logical order (cl. 17) solely based on a *comparing* step. However, the language regarding this 'comparing' teaches comparing <u>each of the</u> lines of the expected code, no mention about whether each line is compared to the generated code. The proper interpretation is <u>that each and all lines of the expected are compared to one another</u>. The disclosure teaches a syntactic tagged format (with enclosed labels) based on which to model code construction, or dictate commands therefor; e.g. matching each of such tagged labels with constructs of the generated code. But nowhere is there explicit description about just comparing each or every line of such special tagged format (i.e. expected code) **to determine** correctness as set forth above. Based on what is clearly disclosed (para 0038, 0044, pg. 11) it is observed that matching of labels sequenced in (**belonging to one line of**) the tagged special command/format (i.e. expected code) with existing constructs of the generated code cannot be construed as comparing *each line of expected code*. The 'comparing each line in the expected code' will be given no weight because it is deemed that the inventor does not possess this limitation at the time the invention was made in order to achieve the *determining* as set forth above. The limitation is treated as though the determining is based on expected syntax derived from a model.

Claim Objections

5. Claim 7 is objected to because of the following informalities: the language recited as 'expected computer code' (cl. 7 line 9: *code that generates an expected computer code*) is deemed a improper terminology or lexicographic representation of the derived information being

gathered from reading a model file (see Specifications: *databases information ... format of the expected code for each block* -para 0027 pg. 7), which are actually disclosed as mere values, inputs, outputs, syntax, all of which not structured (emphasis added) in any code form (see Table pg. 7; *specific information ... is determined* – para 0030 pg. 8; *configuration ... is stored for future use* - top, pg. 9), such as support information to help syntax or formatting (metadata type thereof). Computer code, lexicographically, should be a form that is structured for compilation or for interpretation by a runtime interpreter, whereas meta-information cannot be construed as 'computer code'. The 'expected computer code' will be treated as expected meta-information like syntax rules, input/output parameter requirement … etc.; that is no actual (expected) 'computer code' generation being given patentable weight; and the above language will be treated as generating meta information or requirements to help constructing the actual code. The dependent claims 8-11, 22 for reciting the same language should be corrected accordingly. Appropriate correction is required.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-18, 21-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Charisius et al, USPN: 6,983,446 (hereinafter Charisius)

As per claim 1, Charisius discloses a method for verifying computer code having a plurality of lines generated by a code generating module from a model file of a system including a plurality of functions generated by a model module, the method comprising:

processing the model file (TMM – Fig. 2; Fig. 3; Fig. 13-14), by a verification module (Table 1 → table 18, col. 10 to col. 20; Fig. 19B, 20A – Note: Processing TMM model then invoke a verification tool reads on verification based on processing code from a model),

to determine values, inputs, outputs, function type, and syntax for each of the plurality of functions (e.g. Fig. 21-23; QA audit- Fig. 19B; number of attributes, of import -Table 1, col. 10; Table 3, col. 11; classes … formal parameters, return types, local variables -Table 4 col. 11; Table 11, col. 15; Table 13 col. 17);

generating (refer to USC 112, Rejection) expected code for the generated computer code (Steps 2000-2004, 2008, Fig. 20A, Table 1 –> Table 18, col. 10-20 – Note: Processing TMM model then invoke a verification tool using auditing criteria on naming, declaration, content cohesion, encapsulation coding style **reads on** processing to determine code compliancy, and metrics based on which to enforce language compliancy in terms of programming semantic and syntactic errors or audit violation – *Basic Metrics, Cohesion Metrics, Complexity Metrics, inheritance Metrics, maximum metrics, Ratio Metrics, Style audits, Declaration audits, Name style, Superfluous content*; that is, expected code having syntax or form being language compliant and auditing criteria or code form requirements reads on "expected code")

based on the determined values, inputs, outputs, function type, and syntax for the generated computer code (e.g. number of attributes, of import -Table 1, col. 10; Table 3, col. 11;

classes … formal parameters, return types, local variables -Table 4 col. 11; Table 11, col. 15; Table 13 col. 17; col. 22 lines 40-55);

comparing each line (refer to the USC 112 Rejection regarding *each line* of generated expected code) of the generated computer code and the expected code to determine if the generated computer code includes correct values, correct inputs, correct outputs, correct functions, and correct syntax (e.g. Fig. 4; Fig. 19B-C; Fig. 8B-C; code 1302 – Fig. 13 – Note: actual source code using the SCI reads on *generated code* lines 810, 812, 1302; and audit module to validate expected code being model specifications that encompass rules definitions – audited by metrics - reads on comparing expected and lines of generated; Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; Note: *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A – reads on expected form and syntax being determined in TMM form via template of lines – col. 16, lines 9-14 – and/or metrics as correct values, input/outputs, functions syntax – see Table 1, Table 3, Table 4 from above); and

transmitting an error message if one a line of the generated computer code does not include a correct syntax based on the comparison (e.g. error message … conform to predefined styles; error message when an audit is performed – col. 18 lines 50-55; col. 19 lines 28-34, 52-55; error message - col. 20 lines 25-67).

But Charisius does not explicitly disclose transmitting the error message *if one or more lines of generated code does not include a correct value, correct input, correct output, a correct function*. Using a QA module operating on predefined audit error types as set forth above (Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A –col. 16, lines 9-14 – and/or metrics as correct values, input/outputs, functions syntax – see Table 1, Table 3, Table 4 from above), Charisius discloses error message to be shown to developers regarding many respects, such as syntax, format parameters of a function, declaring of class or members and this is indicative that any violation determined by a verification or audit process to input/output declarations, or correct values or functions would be raised by the validation process. Based on Charisius's and well known practice (in software development) of using a verification/auditing process to systematically notify the developers using visual means on any propriety issue regarding code form or implementation thereof, it would have been obvious for one skill in the art at the time the invention was made to implement the error messages by the source code auditing in Charisius so that error message would be displayed when auditing a line of expected source if the line does not include specifically a correct value, correct input, correct output, a correct function. One of ordinary skill in the art would be motivated to do so because not having such proper value or function, correct input or output thereof a code might not properly compile notably when such type of deficiencies would necessarily fall under the types or style of errors contemplated by a auditing tool based on the above.

As per claim 2, Charisius teaches comparing each line of the generated computer code and the expected code to determine if the generated computer code is missing a line of code (is missing – col 15 lines 47-48; col 25 lines 55-59); and, by virtue of the rationale in claim 1, transmitting the error message if the if the generated computer code is missing the line of code.

As per claim 3, Charisius teaches comparing each line of the generated computer code and the expected code to determine if the generated computer code includes any an extraneous line of code (col. 20 Table 18; col. 22 lines 61-67); and by virtue of the rationale in claim 1, transmitting the error message if the the generated computer code includes the extraneous line of code.

As per claim 4, Charisius discloses comparing each line of the generated computer code and the expected code to determine if the [the] generated computer code is in a logical order (ordered properly – col. 33, lines 55-60 ; col. 34 lines 10-44) and by virtue of the rationale in claim 1, transmitting the error message if the generated computer code is not in the logical order.

As per claim 5, Charisius discloses comparing a first header information section in the generated computer code and second header information section in the expected code to determine if the first header information section matches (*Declaration* – cols. 25-26; match a declaration, col. 37, lines 1-37- Note: generated source code or class package declaration with respect to expected declaration in OO class or Use case package – see Fig. 14-15, 22 -- in an *audit* instance reads on comparing header of a class signature declaration – Note: second header information treated as any form of expected header – see USC 112, 1st para) the second header information section; and (re rationale in claim 1) transmitting the error message if the first header information section does not match the second header information section.

As per claim 6, Charisius discloses comparing a first declared variable section in the generated computer code and a second declared variable section in the expected code (col. 29 lines 7 to col. 30 lines 67; col. 25 lines 20-52 – Note: fixed source code based on positions of declarations, declared type of member with respect to their signature and constructor with respect to a parent class reads on comparing generated variable section with variable section of expected code) to determine if the first declared variable section matches the second declared variable section; and (refer to rationale in claim 1) transmitting the error message if the first declared variable section does not match the second declared variable section.

As per claim 7, Charisius discloses a computer-readable storage medium containing a set of instructions for verifying a generated computer code having a plurality of lines from a code generating module, the generated computer code automatically generated from a model file of a system having a plurality of functions and created by a model module, the set of instructions comprising:

code that reads in the model file (TMM – Fig. 2; Fig. 3; Fig. 13-14;Table 1 → table 18, col. 10 to col. 20; Fig. 19B, 20A – Note: Processing TMM model then invoke a verification tool reads on verification based on processing code from a model);

code that determines values, inputs, outputs, function type, and syntax for each of the plurality of functions in the generated computer code (refer to claim 1);

code that generates (refer to USC 112 Rejection for interpretation of expected code and generating expected code) an expected computer code based on the determined values, inputs, outputs, function type, and syntax (refer to the determining step; number of attributes, of import - Table 1, col. 10; Table 3, col. 11; classes … formal parameters, return types, local variables - Table 4 col. 11; Table 11, col. 15; Table 13 col. 17; col. 22 lines 40-55); code that reads in the generated computer code; code that compares each line in the generated computer code and the expected code to determine if the generated computer code includes the determined values, inputs, outputs, function type, and syntax in the expected computer code (refer to claim 1); and code that transmits an error message *if a first line* in the generated computer code (refer to the USC 112 Rejection) does not include a determined syntax based on the comparison (refer to claim 1).

But Charisius does not explicitly disclose error message if the generated code does not include a determined value, a determined input, a determined output, a determined function based on the comparison. This limitation has been rendered obvious in claim 1

As per claim 8, Charisius discloses wherein the set of instructions further comprises:

code that compares each line in the generated computer code and the expected code (refer to claim 7); and

code that transmits the error message if the first line (refer to USC 112 Rejection) does not include the determined value, the determined input, the determined output, the determined function, the determined syntax, or combinations thereof (refer to rationale in claim 1).

As per claim 9, Charisius discloses wherein the set of instructions further comprises:

code that compares each line in the plurality of lines and the expected code to determine if the plurality of lines includes a second line of code (refer to weight given to *first line, second line* set forth in the USC 112 Rejection) that is not determined in the expected code; and

code that transmits the error message if the generated computer code includes any the second line of code (refer to USC 112 Rejection) not determined in the expected computer code.

As per claim 10, Charisius discloses wherein the set of instructions further comprises: code that compares each line in the plurality of lines and the expected code to determine if the plurality of lines are in a correct logical order (refer to claim 4) and (by virtue of the rationale in claim 1) discloses code that transmits the error message if the plurality of lines are not in the correct logical order.

As per claim 11, Charisius discloses wherein the set of instructions further comprises: code that compares a first header information section in the generated computer code and a second header information section in the expected code to determine if the first header information section matches the second header information section (refer to claim 5); and (by virtue of the rationale in claim 1) discloses code that transmits the error message if the first header information section does not match the second header information section.

As per claim 12, Charisius discloses a system for verifying the contents of a generated computer code having a plurality of lines generated by a code generating module from a model file including a plurality of functions generated by a model module, comprising a processor operable to:

process the model file to determine values, inputs, outputs, function type, and syntax for each of the plurality of functions (refer to claim 1),

generate (refer to USC 112, Rejection) expected computer code for the generated computer code based on the determined values, inputs, outputs, functions type, and syntax for the generated computer code (refer to claim 1),

compare each line in the generated computer code with the expected computer code to determine (refer to claim 1) if the generated computer code includes correct values, correct inputs, correct outputs, correct functions, and correct syntax, and transmit an error message if one a line in the generated computer code (see interpretation of *expected code* in USC 112 Rejection) does not include a correct syntax based on the comparison (refer to claim 1); and a display configured to display the error message (see above), the display coupled to the processor.

But Charisius does not explicitly disclose transmitting the error message if one or more lines of generated code do not include a correct value, a correct input, a correct output, a correct function. This limitation has been addressed in claim 1.

As per claim 13, Charisius discloses wherein the error message indicates if a line is missing in the generated computer code (refer to claim 2).

As per claim 14, Charisius discloses wherein the error message indicates if a line of the generated computer code has any additional content (e.g. *True ... False literals should not be used ... amount of meaningless code ... use of type casts not necessary ... 'abstract' considered obsolete ... should not use parentheses* – col. 19 lines 47 to col. 20 line 23; col. 16 line 58 to col. 17, line 37).

As per claims 15-16, Charisius discloses wherein the processor is operable to compare each line in the expected code (refer to USC 112 Rejection) to the generated computer code to determine if the plurality of lines are in an expected form (Fig. 4; Fig. 19B-C; Fig. 8B-C; code 1302 – Fig. 13; Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; Note: *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A Note: refer to USC 112, 1st paragraph regarding "compare each line in the expected code", i.e. comparing treated as determine language/information derived of a model to determine if such information/language or code is in expected form), and transmit the error message if one or more of the lines of code in the plurality of lines do not match the expected form (e.g. error message … conform to predefined styles; error message when an audit is performed – col. 18 lines 50-55; col. 19 lines 28-34, 52-55; error message - col. 20 lines 25-67);

wherein the processor is operable to compare each line in the expected code (refer to USC 112, 1st para rejection) to the generated computer code to determine if the plurality of lines includes one or more lines of extraneous code (refer to rationale in claim 3).

As per claim 17, Charisius discloses wherein the processor is operable to compare each line in the expected code (Note: comparing treated as comparing model derived constructs or form requirements – see USC 112 rejection) to the generated computer code to determine if each line is in a logical order, and transmit the error message if any line in the plurality of lines is not in logical order (refer to claim 4).

As per claim 18, Charisius discloses wherein the processor is operable to compare a first header information section in the generated computer code to a second header information section in the expected computer code to determine if the first header information section matches the second header information section (refer to claim5), and transmit (by virtue of the rationale in claim 1) the error message if the first header information section does not match the second header information section.

As per claim 21, with reference to claim 1, Charisius discloses steps of:

comparing each line in the generated computer code and the expected computer code to determine if the plurality of lines are complete (col. 16 line 55 to col. 17 line 37; avoid empty catch block … without empty body – Table 17 col. 18-19); and transmitting an error message if the plurality of lines are incomplete (refer to obviousness rationale of claim 1).

As per claim 22, with reference to claim 7, Charisius discloses wherein the set of instructions further comprises code that compares each line in the generated computer code to the expected computer code to determine if the generated computer code is complete (refer to claim 21) and code that transmits the error message (refer to obviousness set forth in claim 1) if the generated computer code is incomplete.

Response to Arguments

8. Applicant's arguments filed 1/22/09 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

USC § 112, 1st paragraph Rejection

(A) Applicants have submitted that the "shorthand notation' are replaced (step 406, Fig. 4) by full data for a header portion, hence the subject matter previously rejected is disclosed. However, the rejection is about not finding support describing how header section of a expected code is generated. The portion regarding changing a Arabic number of a block with its name in terms of filling a <> holder with Sumblock is about replacement that happens in the course of generating code 320 as depicted in the Disclosure (para 0032, pg. 9, step 406); that is, the shorthand notations amounts to a template for representing tagged numeric symbols that would correspond to a model file, in terms of its block numbering and naming, and would help replacing of numeric with names. There is nothing about creating a header that belong to expected code to generated a header section of a expected code source or file. The verification module 102 works with a generated code and constructs derived from a model_file, and information generated from the model (see Table pg. 7; sum blocks 308 - para 0030) are not remotely disclosed as being created then formatted as actual *code* section (including source code lines) preceded by *header* section (including header lines) like in the source code generated in Fig. 3B. The header section of what is recited as 'generated expected code' is not found as being created from step 406. The argument is deemed insufficient to overcome the lack of enablement rejection. The rejection of claims 15-17 is maintained because the Applicants fail to provide factual rebuttals to prove how comparing line of expected code to line of generated code has been disclosed, particularly the comparing is taken in the sense that lines being matched concern with lines of expected code (i.e. generate expected code), a feature which is deemed never properly disclosed by the invention.

USC § 103 Rejection.

(B) Applicants have submitted that Charisius framework is about updating model in conjunction with concurrent and dynamic change in its corresponding source code; and this is not 2 codes being generated as required from 'generating expected computer code' and comparing the generated source code with expected code (Appl. Rmrks pg. 11). The language of 'generating expected code' has been identified as non-enabling and has been treated accordingly. The comparing of code generated with the form or syntax requirements as set forth in Charisius's auditing or code verification process based on the TMM model analysis and source code retrieval is analogized to the 'comparing' as claimed, because 'expected code' has been treated as mere code format, syntax rule or programming language implementation requirement. The claim language has been interpreted based on broad interpretation and contextual insight from the Specifications, about which lack of description is evident. The USC 112 Rejection has provided sufficient grounds as to why the 'expected code' is not being given weight. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference; and the argument is deemed insufficient to overcome the rejection.

(C) Applicants have submitted that the rejection has failed to gather teaching and suggestions from Charisius and OSA to fulfill all the elements of the claim (Appl. Rmrks pg. 12) notably in terms of 'generates an expected computer code based on the determined values, inputs, outputs, function type, and syntax' and 'compares each line … computer code and expected code … if the generated computer code includes determined values … and syntax'. The cited portions have been used to match the above comparing to determine if … includes determined … syntax' and the rationale of obvious was set to render the other items like output, values obvious. The arguments are perceived as mere allegations rather proof of facts pointing exactly where Charisius fail to teach 'generating expected code' and 'comparing … to determine if … proper syntax'; nor are the arguments viewed as clearly establishing factual rebuttals so to point out where in the 103 rationale would there be a negative teaching or real disagreement with Charisius's purpose.

(D) Applicants have submitted that each of the very specific features of claims 2-6, 21, 7-11, 22 and 12-18 are not disclosed or suggested in Charisius' framework in which model-based interface effectuates dynamic upgrade in both modeling view and a source code view as this deficiency has been presented in the argument for claim 1(Appl. Rmrks pg. 12-13). The 'expected code' generation feature is deemed not enabled, and the line of such 'expected code' used in the *comparing* has been interpreted broadly as mere form, syntax requirement; such that Charisius' code audit tool and/or verification criteria are deemed matching the above language or *comparing* step. The argument is referred back to section A, B because it falls under the ambit of the counter-rebut (to Arguments) as set forth therein.

In all, the claims stand rejected as set forth in the Office Action.

Interview Summary

9. The representative, Mr. J. Graff, was very briefly contacted in the first week of April, 09 to the effect of discussing whether some specific feature regarding *shorthand notation*, as initiated by the Examiner, could be implemented into the independent claims in order to build up a claim scenario that would be subject to or candidate for any allowance consideration. But no agreement was ever reached.

Conclusion

10. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

April 17, 2009